

## **Programa Analítico de Python**

### **Modulo I-B Programacion Orientada a Objetos**

1. Introducción al Paradigma POO
2. Crisis del software. Paradigmas de programación: imperativo, lógico y funcional.
3. Antecedentes del paradigma objetos: tipo abstracto de datos.
4. Características de la programación orientado a objetos(POO): encapsulamiento, ocultamiento de la información, polimorfismo y herencia.
5. Solución del paradigma orientado a objetos frente a la crisis: reusabilidad.
6. Terminología POO
7. Clases (moldes), objetos (instancias) y mensajes (comunicación entre objetos).
8. Variables y métodos de clase y de instancia.
9. Superclases y subclases.
10. Clases abstractas.
11. Composición y herencia.
12. Sobrecarga y sobreescritura de mensajes.
13. Binding dinámico vs. binding estático. Persistencia de objetos.
14. Conceptos básicos de Lenguaje Unificado de Modelado.
15. Introducción a Anaconda
16. Entorno de desarrollo, compilación y ejecución. Operadores.
17. Estructuras de decisión y control.
18. Colecciones.
19. Python en el Paradigam de POO
20. Elementos Fundamentales del Lenguaje Python
21. Tipos de datos primitivos.
22. Operadores. Expresiones.
23. Precedencia de operadores.
24. Asignación. Control de flujo de un programa: decisión, repetición y saltos.
25. Clases y objetos.
26. Tiempo de vida y alcance de los objetos.
27. Constructores. Variables y métodos de clase y de instancia.
28. Sobrecarga de métodos.
29. Finalización de objetos y Garbage Collection.
30. Construcción de un programa en Python.
31. Manejo de Secuencias.
32. Ejemplos de clases existentes: System y String.
33. Clases anidadas. Interfaces: su implementación.
34. Violación del paradigma POO: acceso a variables de instancia públicas desde afuera de la clase.
35. Herencia
36. Sintaxis de la herencia en Python.
37. Orden de invocación de constructores.
38. El modificador final.
39. Sobre-escritura de métodos vs. Sobre-carga de métodos.
40. Clases abstractas.
41. La clase Object.
42. Jerarquías de clases existentes en Python.
43. Ejemplos de clases: Random, Stack, Vector y Hashtable.

- 44. Excepciones
- 45. Manejo de errores.
- 46. Excepciones básicas.
- 47. Captura y manejo de excepciones en Python. Construcción de programas seguros.
- 48. Sistema de Entrada/Salida
- 49. Revisión del concepto de flujo de datos: stream.
- 50. Clases para el manejo de streams.
- 51. Persistencia de objetos.
- 52. Pruebas de unidad
- 53. Diseño orientado a pruebas.
- 54. Interface con el usuario.
- 55. Interface gráfica.

# Programa Analitico de Python

## Modulo I-B Programacion Orientada a Objetos

### 1 Modulos de Aplicaciones

- 1.1 Modulo Random
  - 1.1.1 Funciones de Distribucion
- 1.2 Modulo Statistics
- 1.3 Modulo Array
- 1.4 Modulo bisect
- 1.5 Modulo heapq

### 2 Clases y Objetos

- 2.1 Objetos
- 2.2 Clases
  - 2.2.1 Instantiation (Construction)
- 2.3 Funciones con Atributos
- 2.4 pass : Place Holder
- 2.5 Overloading (Sobrecarga)
  - 2.5.1 Overloading en Python basico
- 2.6 Inheritance
  - 2.6.1 Multiple Inheritances
- 2.7 super()
- 2.8 Execution from Command Line

### 3 Time Related Operations

- 3.1 Time Standards
- 3.2 time Module
- 3.3 datetime Module
  - 3.3.1 time Objects
  - 3.3.2 datetime Objects
  - 3.3.3 Time Intervals
  - 3.3.4 tzinfo
  - 3.3.5 Algebra with Time Objects
- 3.4 Calendars
- 3.5 timeit Module

### 4 Ayudas para Programacion Funcional

- 4.1 Modulo operator
  - 4.1.1 Metodos Genericos
  - 4.1.2 Operadores inplace
- 4.2 itertools
  - 4.2.1 Filtering.
- 4.3 generator Using yield
- 4.4 iterator Formation
- 4.5 decoratorS
- 4.6 functools.
  - 4.6.1 total\_ordering
  - 4.6.2 single dispatch Generic Function.

4.6.3 Objetos Parciales

4.6.4 Funciones de Reduccion